

High availability infrastructures for TYPO3 Websites

Luc de Louw

bit-heads GmbH, Internet Services, Rosengartenstrasse 72,
CH-8037 Zürich, Switzerland.
luc.delouw@bit-heads.ch

Abstract. The best content management software, the best design and the best content is useless if the server is not reachable. Today companies everywhere depend not only on maximum uptime of in-house applications, but also on the continuous availability of websites and web-based services, many of them programmed with TYPO3. The goal of this paper is to introduce high availability (HA) clustering as a practical, cost-effective option for enterprises seeking continuous operations of web applications, and more generally as a solution anywhere the business cannot tolerate a disruption in processing due to a server failure. Moreover we introduce the technology DRBD (Distributed Replicated Block Device).

1 Introduction

As websites and web based services are used more widely for mission critical business applications, support for high availability through application failover is becoming more important. Interruptions or delays in service impair the delivery of enterprise services and harm customers and suppliers. High availability clusters allow the application and business process to resume operations quickly and ensure that business is not interrupted.

Improving availability of the IT infrastructure involves employing both hardware and software technologies. Redundancy throughout the system infrastructure ensures there is no single point of failure.

The paper contrasts available technologies for HA clusters. The focus is on providing the solution to achieve scalable performance and high availability at low cost, as delivered by the technology DRBD (Distributed Replicated Block Device).

Furthermore, the paper shows how asynchronous mirroring helps mitigate the effects of a major disaster in a data center, i.e. flooding, fire, earthquakes and offers quick recovery from such an event.

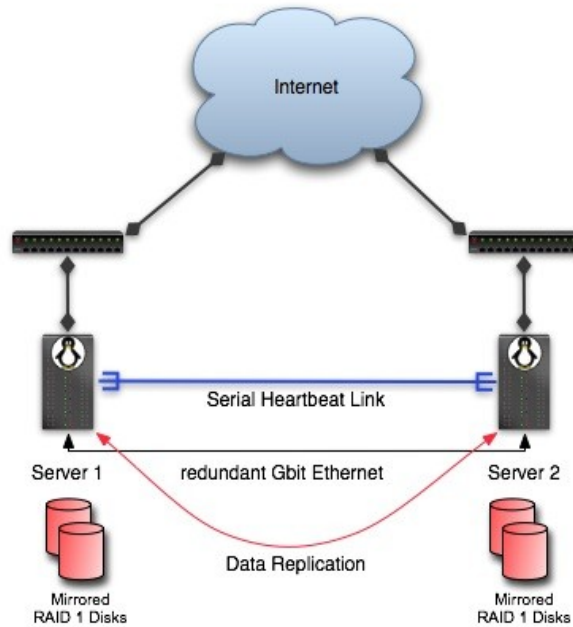


Figure 1: Scheme of a HA-Cluster

Figure 1 shows the schematic picture of a HA-cluster. Further below in the paper the parts and connections will be explained in more details.

1.1 Why using High availability infrastructures

Businesses large and small can benefit from high availability clusters. The benefits directly improve the bottom line. They reduce business interruptions due to system failure, they boost productivity through increased uptime and they avoid customer frustration and dissatisfaction over loss of service. Furthermore, the low cost of high availability solutions as described in this paper reduces the upfront investment and the total cost of ownership (TOC).

There are HA solutions available that have been specifically developed for use with today's commodity hardware products; they do not require expensive, special-purpose hardware or software components from Sun, IBM and other industry leaders.

In addition to reduced investments in hardware, there is a significant cost saving potential on the operating level.

Due to redundancy, there is no need to repair a failed server immediately, thus expensive maintenance service level agreements with server vendors can be adjusted. The same applies to software problems. The passive server automatically takes over the services.

Hardly any outage is caused by planned maintenance work such as kernel updates which need a reboot. The service interruption strongly depends on the application to be started. The typical downtime for a default TYPO3 installation is just some four seconds.

The two nodes of the cluster can run different versions of a software, which allows testing the compatibility of a new version with the existing data without interrupting the business process.

1.2 Who is using it?

Solutions as described in this paper make high availability a practical option for almost any enterprise of any size. There is no need for special hardware or software. This means that even small and mid size organizations can deploy high availability systems today to achieve the same level of predictable, consistent service that large organizations with costly complex infrastructure deliver. For these organizations, this means increased productivity and high customer satisfaction with a simplicity that makes it practical and a price that makes it affordable.

For a selection of clients implementing the solution presented in this paper, please visit www.linbit.com as well as the certified partner sites such as www.bit-heads.ch.

1.3 Identify and eliminate single points of failures

Enterprise-strength clusters are configured to have no single points of failure, by including redundancy at all levels. Most hardware failures are bound to the same components. Redundant installation of components prevents these failures. The following components need careful attention:

- Hard disks
- Power supply
- Cooling
- Network connectivity

The **hard disk** is the weakest point in a server. The micro mechanical parts in the hard disk are very sensitive to heat and vibrations. Another problem of the hard disks are the bearings with their limited lifetime. Installing a RAID 1 (mirroring) hard disk controller to replicate data across multiple local disks is needed. A different disk

producer for the second disk is recommended. Redundant disks handled by a RAID controller are further called logical local disks.

The next priority is the continuity of the **power supply**. By installing redundant power supply systems connected to different power lines, uptime is unaffected by power loss on a single line. In addition, redundant battery backed UPS (Uninterruptible Power Supply) systems are recommended to cover power loss affecting all power lines in the data center.

CPUs and other components need active air **cooling**. Installing additional fans greatly reduce the danger of overheating components such as CPUs and hard drives. Modern servers allow to monitor the speed of fans and alert the system administrator when the fan speed falls below a certain threshold.

A rare but possible event is the failure of **network connectivity** due to failure of a switch port or a complete switch. Channel bonding circumvents a single switch port failure. More sophisticated is the bonding of Ethernet interfaces with a connection to two switches. Such a redundant connection to two switches needs a redundant ISL (Inter Switch Link) between the two switches involved and switches that are capable of running the STP protocol (Spanning Tree Protocol) to avoid switching loops. A single managed switch such as the Cisco Catalyst with redundant power supply will usually be good enough and keep things simple.

1.4 Introduction to HA-clustering

There are three different types of clusters: High availability clustering (HAC), high performance computing clustering (HPC) and load leveling (load balancing) clusters. This paper focuses on the HAC method of clusters.

Understanding how applications and data can be distributed across multiple servers in a cluster is important. In the Active/active clustering, both nodes are actively running a share of the total load. In the Active/passive clustering, only one node is productive.

1.4.1 Different data storage methods

In data storage the alternatives are shared dual end SCSI versus distributed data with real time synchronization. Installing shared SCSI components costs as much as five times more money than using the distributed data approach and is less reliable in a split-brain situation since most SCSI controllers do not have fencing support to prevent the disks mounting on both nodes. A split brain situation occurs when the nodes are not able to communicate with each other and both are bringing up the IP addresses and services. The conclusion is, that simple raid 1 arrays with SATA disks

on each node with real-time synchronization is more reliable and cost effective, while offering almost equal performance to shared dual end SCSI.

1.4.1 Active/passive vs. active/active clustering

In the event that the customer wishes to run a single large application on the cluster, it must be remembered that servers cannot access the same disk partition at the same time (few applications today provide support for concurrent data update from multiple systems). So, it is necessary to restrict these applications to a single server, leaving the other server as ready-to-go backup in the case of failure. This is called Active/passive operation. This style of operation keeps maintenance at a low level, however it leaves the passive system idle, which can be a waste of valuable computing power.

The alternative is active/active clustering. At the time of writing, active/active clustering is only possible with shared SCSI disks. This will change in the near future with the release of DRBD8 and the usage of OCSF2. With such a future configuration web servers can run on both nodes at the same time. Some databases like MySQL provide support for clustering as well.

In conclusion, Active/active clustering is much more complex, resulting in more costly maintenance. At the time of writing, it is not the recommended solution. Active/passive configuration are easier to maintain and less complex.

1.4.2 Monitoring and automated fail over

Essential to a high availability cluster is the software which monitors the availability of the servers in the cluster. There are many different software solutions available, however most of them are highly proprietary and expensive. Commercial examples are HACMP from IBM as well as HP Service Guard. HACMP is available for IBM AIX and Linux. HP Service guard is available for HP-UX and Linux.

The basic functionality of the software is monitoring the partner node and switch over in case of a detected failure. Each server heartbeats or “pings” the partner node with regular short messages to check that they are operating correctly. If a series of retried heartbeats do not receive an acknowledgment then a server will assume that the remote server has failed and trigger a cluster transition, which removes the failed server from the configuration and initiates application service failover. The measures taken include the takeover of the service IP addresses, the mount of the file systems and the start of the applications needed for production.

2 A closer look at DRBD and Heartbeat

The solution for implementing a software-only based HA cluster is DRBD and heartbeat as the monitoring software. It is the most reliable open source solution available on the market.

2.1 How DRBD works

DRBD is a Linux module which adds the capability of real time replication of data from a local block device to another block device in a remote node.

2.1.1 DRBD is a Linux only software

DRBD is a Linux kernel module that adds an additional layer between the disk driver and the file system. Linux kernel modules are specifically built for a particular kernel and thus only available for Linux.

2.1.2 The software stack

The first layer in the software stack is the disk driver such as SCSI or SATA. Typically, a SCSI hard drive on the Linux system is in the name space /dev/sd*. A SATA or PATA disk is on /dev/hd*. These devices are the underlying devices of DRBD. The DRBD device names are /dev/drbd*, i.e /dev/drbd0.

DRBD's underlying devices can be any kind of block device such as whole disks, disk partitions, logical volumes, or even floppy disks.

The file system is agnostic to the type of device in the underlying layer, because DRBD behaves like any other block device. This makes DRBD completely application independent.

2.1.3 Data replication

DRBD replicates the data in real time by block. Each write-access to the local logical disk triggers a network operation to write the affected blocks on the secondary node as well. The result is that both nodes have the same data stored on its local disks.

DRBD uses three different protocols with different levels of reliability. Protocol "A" ensures only the write commit on the local drive of the primary node and is sufficient for asynchronous mirroring of data. Protocol "B" adds a level of reliability, namely

write commit on the local node happens only if the remote nodes receive buffer received the data.

The most reliable protocol is “C”. Local write commits only happen if the data is also completely stored on the remote node. HA-clusters mainly use protocol “C” because it guarantees transaction safe file system operations. However, protocol “C” needs a high bandwidth network between the two nodes. Protocol “A” and “B” are used if the bandwidth between nodes is limited.

For a reliable and high performance service, the HA-cluster node must be located in the same data center and running protocol “C”.

It is strongly recommended to use a separate Gbit Ethernet interface for the DRBD replication. Typically this is done by a direct connection via cross over Ethernet between the two nodes.

However, the drawback of protocol “C” is the slower throughput of the local logical disk compared to writing to native disk or using protocols “A” or “B”. DRBD offers tuning options that reduce the loss of speed to approx. 3%.

2.2 Installing DRBD

In advance to installing DRBD, the system must be set up with the Linux kernel sources as well as the GCC compiler.

Assuming the above mentioned set up is provided, continue as following:
Unpack the DRBD distribution and issue:

```
make && make install
```

The installation process includes the installation of an init script for the used Linux distribution.

Some Linux distributions such as Suse Linux or Debian Sarge are providing binaries that are easier to install. It is recommended to consult the distribution manuals for more information. If the distribution does not provide binary packages for DRBD, refer to the LINBIT website <http://www.linbit.com/support/drbd-current/>

2.3 Configuration of DRBD

There are only a few things to configure in the configuration file, such as the underlying device for DRBD. This is done by editing the default configuration file.

The configuration file of DRBD is straight-forward, The following must be configured:

- The underlying device
- What kind of protocol is to be used
- The connectivity parameters such as IP addresses and ports.

Assuming that the partition dedicated to DRBD is on both nodes /dev/hda2 the following DRBD configuration is to be used.

The configuration file is located in /etc/drbd.conf

```
resource r0 {
    protocol C;
    startup {
        degr-wfc-timeout 120;    # 2 minutes.
    }

    disk {
        on-io-error    detach;
    }

    syncer {
        rate 20M;
        al-extents 257;
    }

    on node-one {
        device    /dev/drbd0;
        disk      /dev/hda2;
        address   192.168.100.1:7788;
        meta-disk internal;
    }

    on node-two {
        device    /dev/drbd0;
        disk      /dev/hda2;
        address   192.168.100.2:7788;
        meta-disk internal;
    }
}
```

Figure 2: DRBD configuration file

The “resource” name can be defined by the system administrator, named for example typo3_0. For the choice of the protocol please see above. In the stanza “startup” the initial wait time on a degraded cluster must be defined, i.e. the time the nodes are waiting for a connection to its partner node.

In the stanza “syncer” the maximum speed of (re)-sync of the data between the two nodes must be defined. The value depends on the maximum network speed and the native disk speed on the two nodes.

Very important is the stanza of “node-one” and “node-two”. These names are reflecting the effective node names and must be equal to the output of “uname -n” as seen in Figure 2.

A possible pitfall is an installed and activated firewall. Please ensure that the two nodes are able to communicate with TCP port 7788.

3 The role of heartbeat

Heartbeat has a central role in HA-clustering. It is the software that monitors the active node and waits for its failure. If the active node is not responding within the defined time parameter, heartbeat takes over the IP addresses, disks and services.

Since the release of heartbeat version 2.x it is possible to configure the cluster with a XML-based configuration and status file. At the moment this method is considered stable but adds much more sources for mistakes because of the complexity of the file. Furthermore, the cluster configuration and the status information are in the same file, which means that this important file is being changed during runtime. This is not considered a good programming style. Therefore it is strongly recommended to use the old-style version 1.x method of heartbeat configuration as described below.

3.1 Installing and configuring heartbeat

There are three important files for the configuration of heartbeat: ha.cf, haresources and authkeys.

All of these files are located in /etc/ha.d. ha.cf contains the information such as node names and heartbeat connections. The file haresources contains the definition of the services in a cluster. Authkeys is the file containing a shared secret preventing unauthorized communication.

The following is an example of a shortened ha.cf file:

```
auto_failback off
ucast eth0 10.0.0.1
ucast eth1 192.168.0.1
serial /dev/ttyS0

node node-one
node node-two
```

Figure 3: Heartbeat ha.cf configuration file

The “authkeys” file contains a shared secret which is encrypted by a defined algorithm.

These are the only values that need to be changed after installation.

The parameter “auto_failback off” tells heartbeat not to switch back to the default node after the recovery of a failed node. This makes sense if both nodes are fast enough to support production. However, if an older server with reduced performance is your second node, you should leave the parameter in the default value “on”.

The default is to communicate over multi cast IP addresses (mcast). Using multi cast IP addresses has the benefit of being able to have the same configuration file on both nodes. The drawback in the case you are operating multiple HA-Clusters in the same subnet is, that other nodes in that subnet are able to pick up the communication. If you have multiple clusters in the same subnets, heartbeat will get confused and heartbeat’s functionality can be impacted negatively. A smarter solution is using unicast communications. With unicast communication however the configuration files on both nodes need to be different from each other. The example as outlined above assumes that there are two Ethernet interfaces installed. One of them is the public network, here represented by 10.0.0.0 and a direct connection via cross over Ethernet in the network, represented by 192.168.0.0, which is also used by DRBD for the replication. With the deployment of a serial link there is the additional benefit of avoiding a split brain situation. If the IP stack or both network connections will break down, heartbeat is still able to communicate through the serial line.

At the end of the configuration file the node names involved have to be defined. It is essential that the node names defined here are equal to the output of “uname -n”. The first node defined is the default node.

The configuration file “haresources” is used to define the file systems, IP addresses and applications to be handled by the cluster. It is essential that this file is exactly the same on both nodes.

The default configuration file needs the following changes:

```
node-one 10.0.0.3 192.168.0.3 \  
    drbddisk::r0 \  
    Filesystem::/dev/drbd0::/data1::ext3 \  
    mysql.server \  
    apachectl
```

Figure 4: Heartbeat haresources configuration file

“node-one” is representing the default node, but it is valid for both involved nodes, followed by the definition of the services IP addresses. These IP addresses are active on the respective active node and are shared between both nodes. As many IP addresses as needed can be configured. In the example above, two service IP’s are defined, one on the public network and one on the private network.

All production services such as apache etc. must be configured to use these service IP addresses.

If there is more than one parameter used, parameters are separated by two colons. All configuration parameters are either on a single line or alternatively, lines are separated by backslash.

The next line defines the involved DRBD device. In the example above, heartbeat will be responsible to handle the resource “r0”. The next line is the instruction which device should be mounted to which mount point and which file system type is to be used.

The next lines are the applications to be started on fail over. “mysql.server” and “apachectl” are scripts that can either be located in /etc/init.d or /etc/ha.d/resources.d. At the start of these scripts, heartbeat automatically adds the parameter “start” and “stop” on stopping the services on a switch over.

Most modern Linux distributions are providing a rpm or deb package for easy installation. Please consult your distributors manual for details.

3.2 Use STONITH devices to definitively avoid split brain situations

The worst case in a production HA-Cluster is a split brain situation. This happens if all of the multiple redundant communication channels for heartbeat communication are failing. In such a case, both of the nodes assume the other node dead and bring up the service IP addresses, mount the file systems and start the services. This renders services inoperative, inconsistent data is possible. It is very unlikely that this happens when using two Ethernet interfaces plus a serial link for the heartbeat communication.

The only way to completely prevent such a situation is the installation of so called STONITH devices (Shoot The Other Node In The Head). Such a device can be a UPS powering the other node. Heartbeat has a interface for such UPS devices. The result is a race game, the node which first detects the other node dead will power-off the competing node. Because split brain situations are very unlikely, it will not be further described in the paper.

3.2 Manual switch over

If there is a need to manually switch the active node, a manual switch over can be initiated. Depending on your production applications, the time such a switch takes strongly depends on your application. For a typical TYPO3 installation it takes about four. To initiate a controlled switch over, issue:

```
heartbeat standby
```

As a result, all applications will be properly shut down, file systems will be unmounted while on the previously passive node the respective services will be started.

4. Disaster safety with DRBD+

As outlined above, the two redundant nodes need to be located in the same data center due to safety and performance reasons. However, both nodes will be impaired by a disaster affecting the data center such as fire, flooding, etc. and data loss cannot be prevented.

Traditional solutions include the off-site backup of the data. However, recovering the data from the off-site location and the setup of new servers capable of running in the production mode is time consuming and depends on your applications and the amount of the data. This causes severe interruptions to the business processes, and harms customers and suppliers.

Alternatively, DRBD+ is the recommended solution preventing business interruptions due to disaster recovery. DRBD+ is a commercially available solution based on DRBD that provides companies with a third, asynchronous mirroring to a third server over an unlimited distance. DRBD+ is using protocol "A" to the remote location, while still using protocol "C" locally between the two main nodes. As outlined above, using protocol "A" for the off-site mirroring has some drawbacks. Worst case is the need of running a file system check and a MySQL check on the databases to repair corrupted data. However, a major data loss is not expected.

The third asynchronous mirror is outside of the control area for the heartbeat software. In the case of a disaster, manual switch over to the remote data center has to be done. Both the file systems as well as the applications need to be mounted manually.

Typically, the service IP addresses cannot be taken over. The recovery is handled with a change of the DNS entries, meaning the TTL (Time To Live) of the DNS entries of the servers need to be low. The TTL needs to be as low as the service availability interruption is defined in the service level agreement.

5. Conclusions

With the use of DRBD and heartbeat based software-only HA-clusters it will be possible to deliver a complete solution based on open source software. This begins on the operating system level and ends on the application level. A typical setup could look like this:

- Linux operating system
- Apache web server
- MySQL database
- PHP server side scripting language
- TYPO3 enterprise content management system

Delivering a complete open source stack has many benefits for companies of every size. A short (incomplete) list of the benefits of using open source software is:

- Access to the complete source code
- Source code can be modified to individualize the software
- Independence from a particular software vendor
- Security problems will be detected and solved earlier

Companies that require protection of their data and continuation of their operations even in a disaster scenario should evaluate the DRBD+ solution. The annual license fee finances the further development of DRBD.

6. Further readings

<http://www.linux-ha.org> is a comprehensive resource for HA-clusters on Linux as well as the provider for downloading heartbeat. The project is active and growing for years now. The wiki on that website is large, but may be confusing from time to time. The people on the mailing list have the reputation of being friendly and helpful.

<http://www.drbd.org> is the DRBD project website operated by LINBIT, the company managing DRBD. There is a library of papers written mostly by Philipp Reisner, author of DRBD.

7. Acknowledgments

Christian Wessalowski for proofreading of the paper
Claudia Nussdorfer for content review and hints.